IN THE CLAIMS

1. (Original) A method comprising:

receiving a first source code having a number of global storage objects, wherein the number of global storage objects are to be accessed by a number of threads during execution; and

translating the first source code into a second source code, wherein the translating includes adding initialization logic for each of the number of global storage objects, the initialization logic to include the following:

generating private copies of each of the number of global storage objects during execution of the second source code; and

generating at least one cache object during the execution of the second source code, wherein the private copies of each of the number of global storage objects are accessed through the at least one cache object during execution of the second source code.

2. (Original) The method of claim 1, wherein the at least one cache object includes a number of pointers, wherein each of the pointers points to a private copy of a global storage object for a thread.

3. (Original) The method of claim 1, wherein a private copy of a global storage object for a thread is accessed through the at least one cache object, independent of a run time library, after the private copy has been generated.

4. (Original) The method of claim 3, wherein the private copy of the global storage object for the thread is generated through execution of a routine of the run time library.

5. (Original) The method of claim 1, wherein the private copy of the global storage object for the thread is generated through execution of the second source code, independent of the run time library.

6. (Original) The method of claim 1, wherein the first source code and the second source code can be executed across at least two different platforms.

7. (Original) The method of claim 1, wherein the first source code and the second source code can be in at least two different programming languages.

8. (Original) The method of claim 1, wherein the second source code is to execute in a multi-processing shared memory environment.

9. (Original) The method of claim 1, wherein generating the at least one cache object during the execution of the second source code comprises creating the at least one cache object through an invocation of a routine within a run time library upon determining that the at least one cache object has not been generated.

10. (Original) The method of claim 9, wherein the initialization logic comprises receiving a pointer to the at least one cache object and the pointer to the private copy of the global storage object for the thread from the routine within the run time library.

11. (Original) A method comprising:

receiving a number of program units having a number of global storage objects, wherein the number of global storage objects are to be accessed by a number of threads during execution in a multi-processing shared memory environment; and

translating the number of program units into a number of translated program units, wherein the translating includes adding initialization logic for each of the number of global storage objects , the initialization logic to include the following:

generating thread private copies of each of the number of global storage objects for each of the number of threads during execution of a routine from a run time library, the thread private copies of each of the number of global storage objects generated by a routine in a run time library; and

generating at least one cache object during execution of the routine from the run time library, wherein a thread private copy of each of the number of global storage objects are accessed through the at least one cache object during execution of the second source code, independent of the run time library, after the thread private copy has been generated.

12. (Original) The method of claim 11, wherein the at least one cache object is stored in a software cache for the number of program units during execution of the translated program units.

13. (Original) The method of claim 11, wherein the at least one cache object includes a number of pointers, wherein each of the number of pointers points to a private copy of a global storage object for a thread.

14. (Original) The method of claim 11, wherein the initialization logic comprises receiving a pointer to the at least one cache object and the pointer to the thread private copy of the global storage object for the thread from the routine within the run time library.

15. (Original) The method of claim 11, wherein the first source code and the second source code can be executed across at least two different platforms.

16. (Original) The method of claim 11, wherein the first source code and the second source code can be in at least two different programming languages.

17. (Original) A system comprising:

a translation unit to receive a number of program units having a number of global storage objects, wherein the number of global storage objects are to be accessed by a number of threads during execution in a multi-processing shared memory environment, the translation unit to translate the number of program units into a number of translated program units, wherein the number of translated program units are to generate at least one cache object and to generate thread private copies of each of the number of global storage objects for each of the number of threads during execution, the thread private copies of each of the number of global storage objects generated by a routine in a run time library, wherein the thread private copies of the number of global storage objects are subsequently accessed through the at least one cache object, independent of routines in the run time library; and

a compiler unit coupled to the translation unit, the compiler unit to receive the number of translated program units and to generate object code based on the number of translated program units.

18. (Original) The system of claim 17, comprising an execution unit coupled to the translation unit, the compiler unit and the run time library, the execution unit to receive the object code and to execute the object code in a multi-processing shared memory environment.

19. (Original) The system of claim 17, wherein the first source code and the second source code can be executed across at least two different platforms.

20. (Original) A machine-readable medium that provides instructions, which when executed by a machine, cause said machine to perform operations comprising:

receiving a first source code having a number of global storage objects, wherein the number of global storage objects are to be accessed by a number of threads during execution; and

translating the first source code into a second source code, wherein the translating includes adding initialization logic for each of the number of global storage objects , the initialization logic to include the following:

generating private copies of each of the number of global storage objects during execution of the second source code; and

generating at least one cache object during the execution of the second source code, wherein the private copies of each of the number of global storage objects are accessed through the at least one cache object during execution of the second source code.

21. (Original) The machine-readable medium of claim 20, wherein the at least cache object includes a number of pointers, wherein each of the pointers points to a private copy of a global storage object for a thread.

22. (Original) The machine-readable medium of claim 20, wherein a private copy of a global storage object for a thread is accessed through the at least one cache object, independent of a run time library, after the private copy has been generated.

23. (Original) The machine-readable medium of claim 22, wherein the private copy of the global storage object for the thread is generated through execution of a routine of the run time library.

24. (Original) The machine-readable medium of claim 20, wherein the private copy of the global storage object for the thread is generated through execution of the second source code, independent of the run time library.

25. (Original) The machine-readable medium of claim 20, wherein generating the at least one cache object during the execution of the second source code comprises creating the at least one cache object through an invocation of a routine within a run time library upon determining that the at least one cache object has not been generated.

26. (Original) The machine-readable medium of claim 25, wherein the initialization logic comprises receiving a pointer to the at least one cache object and the pointer to the private copy of the global storage object for the thread from the routine within the run time library.

27. (Original) A machine-readable medium that provides instructions, which when executed by a machine, cause said machine to perform operations comprising:

receiving a number of program units having a number of global storage objects, wherein the number of global storage objects are to be accessed by a number of threads during execution in a multi-processing shared memory environment; and

translating the number of program units into a number of translated program units, wherein the translating includes adding initialization logic for each of the number of global storage objects , the initialization logic to include the following:

generating thread private copies of each of the number of global storage objects for each of the number of threads during execution of a routine from a run time library, the thread private copies of each of the number of global storage objects generated by a routine in a run time library; and

generating at least one cache object during execution of the routine from the run time library, wherein a thread private copy of each of the number of global storage objects are accessed through the at least one cache object during execution of the second source code, independent of the run time library, after the thread private copy has been generated.

28. (Original) The machine-readable medium of claim 27, wherein the at least one cache object is stored in a software cache for the number of program units during execution of the translated program units.

29. (Original) The machine-readable medium of claim 27, wherein the at least one cache object includes a number of pointers, wherein each of the number of pointers points to a private copy of a global storage object for a thread.

30. (Original) The machine-readable medium of claim 27, wherein the initialization logic comprises receiving a pointer to the at least one cache object and the pointer to the thread private copy of the global storage object for the thread from the routine within the run time library.